Power Management Techniques in an FPGA-Based WSN Node for High Performance Applications

M. Lombardo, J. Camarero, J. Valverde, J. Portilla, E. de la Torre, T. Riesgo
Centre of Industrial Electronics, Technical University of Madrid
José Gutiérrez Abascal, 2; 28006 Madrid, Spain
eduardo.delatorre@upm.es

***Abstract*—In this work, the power management techniques implemented in a high-performance node for Wireless Sensor Networks (WSN) based on a RAM-based FPGA are presented. This new node custom architecture is intended for high-end WSN applications that include complex sensor management like video cameras, high compute demanding tasks such as image encoding or robust encryption, and/or higher data bandwidth needs. In the case of these complex processing tasks, yet maintaining low power design requirements, it can be shown that the combination of different techniques such as extensive HW algorithm mapping, smart management of power islands to selectively switch on and off components, smart and low-energy partial reconfiguration, an adequate set of save energy modes and wake up options, all combined, may yield energy results that may compete and improve energy usage of typical low power microcontrollers used in many WSN node architectures. Actually, results show that higher complexity tasks are in favor of HW based platforms, while the flexibility achieved by dynamic and partial reconfiguration techniques could be comparable to SW based solutions.**

Keywords: FPGA based WSN node; low power design; partial reconfiguration; power islands, energy management.

I. INTRODUCTION

WSN applications are evolving towards more demanding scenarios and requirements. This fact implies the use of much more powerful processing units to be able to deal with new algorithms and higher amounts of data.

In classic WSN applications, simple tasks have been always performed by ultra-low power microcontrollers with very limited computing capabilities. However, a growing tendency to higher compute demanding applications has appeared. Some factors that characterize these new applications are: a) the use of more complex sensors such as cameras, radars or ultrasound positioning equipment, among others; b) intensive processing tasks running either autonomously in every node or as part of the network management, c) the increase of raw data to be processed due to the use of these more complex algorithms or to the increase in the number of nodes, etc..

Even though the use of FPGAs in WSN applications has been avoided by many designers mostly because of their high power consumption, results show that by taking advantage of HW acceleration together with some power management techniques, it is possible to obtain energy efficient solutions that are suitable for these high performance WSN applications.

As it will be explained within the next sections, the typical application profile of a WSN has, in general, a very low duty cycle, see Figure 1.

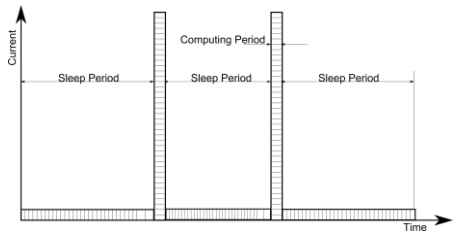

Figure 1: Typical WSN application profile

Therefore, it is crucial for these nodes to have extremely low power consumption during sleep time, since the node will be most of the time in this mode. This may be achieved by either selecting devices with very low static power consumption or, if not possible, to implement the required power on/off control for some components.

Although SRAM-based FPGAs cannot compete in terms of static power consumption with other FPGA technologies, like the non-volatile ones, the increased flexibility provided by the possibility of dynamically and partially reconfigure them, as well as the higher resource availability, may make these RAM based FPGAs good candidate solutions for high performance applications. In this way, the purpose of this work is to show the power management techniques that are applied to a custom designed node architecture with an SRAM FPGA, so that the main drawback of excessive energy consumption is minimized.

One problem of RAM-based FPGAs is their relatively high static power consumption even when activating their power save modes. Another drawback is that they are not alive at power up, so every time the FPGA is powered off, its configuration is lost. Due to these reasons, in order to minimize power consumption during sleep time, the FPGA must be switched off so that it has to be configured again after every off cycle.

In Figure 2, a typical power consumption profile of the RAM-based FPGA node is shown. Three regions are differentiated: sleep period, reconfiguration period and computing period. These plots, since voltage is kept constant,

may be obtained by measuring current consumption, as it will be done in the experiments shown at the end of the paper.

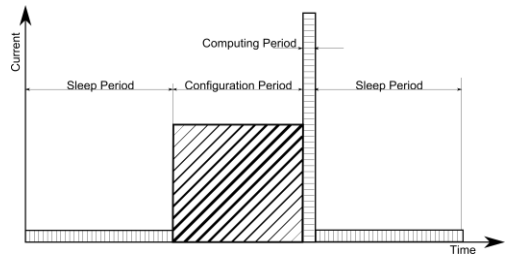

Figure 2: Energy profile of the High Performance Node

The power management techniques described in this paper focus on the reduction of the current consumption during these three periods. The current consumed during sleep periods is reduced by switching on only those components that are required to determine the wake-up conditions. The configuration time, as it will be shown will be reduced by diminishing the reconfiguration time, since as we have observed within the experiments current consumption is roughly independent on the type of bitstream. Finally, consumption reduction during the computing period is exercised by moving most tasks from SW into HW, with some energy saving improvements which are the main results published by the authors in [1].

II. Related Work

High performance applications are often related to the use of complex algorithms such as video compression, data encryption, tracking, etc. Many of them are multimedia applications that include low-power video cameras or microphones, and they are known as Wireless Multimedia Sensor Networks (WMSN). In [2] and [3], surveys about these applications are detailed. According to [3], these applications are classified in: surveillance, traffic monitoring and enforcement, personal and health care, gaming and environmental and industrial. Surveillance is probably the best example of the increase of complexity in traditional WSN applications. Traditionally, surveillance based on WSNs was limited to intruder detection or movement in target areas. Features like cameras or node synchronization permit location tracking or people identification, as seen in [4], [5], [6] and [7].

The use of these capabilities can also make a breakthrough in terms of human health-monitoring applications, mostly related to telemedicine and complete patient monitoring. Also, non-intrusive study of people behavior, mainly elder people suffering dementia, has been reported in [8] and [9].

Applications related to environmental care and industrial monitoring can be also faced and improved by means of using these high performance networks. For instance, full manufacturing processes including quality control can be monitored relying on artificial vision techniques.

Not only the inclusion of new sensors but also the toughening of traditional constraints such as maximum latency, bandwidth, the increase in the number of nodes or security requirements, causes architectural changes in WSN platforms. In [10] and [11], a survey about this kind of applications is studied including encryption algorithms ([12] [13]) which at the beginning where considered unfeasible to be carried out by WSNs. As an example of complex data calculations, in [14], the authors face data-mining for WSNs while in [15] distributed multimedia source coding is addressed.

Using FPGAs in wireless sensor nodes for high demanding scenarios is not a novel approach. In [16], for instance, the authors introduce a Spartan 3E prototype board as a coprocessor attached to an external ZigBee transceiver for the implementation of a hyper-chaos encryption engine. Similar applications using off-the-shelf FPGA boards are shown in [17] [18] [19] and [20]. These approaches prove that including hardware-based devices in WSNs offer benefits in terms of flexibility and performance. However, in spite of being valid for proof of concept, existing solutions are far from showing real new WSN architectures, leaving behind important aspects such as power consumption, power management, sensor integration. On the contrary, the development of a complete FPGA based node is provided in [21], including a low-performance Spartan 2E. The complete node including communications is integrated in a 25 mm × 25 mm board.

Hardware reconfigurability is addressed for WSN nodes. In [22], Philipp and Glesner propose a virtual reconfiguration layer on top of a flash-based low power FPGA, which is the main computing device, with no microcontroller aside. This approach enhances notably the flexibility by allowing HW changes, but limits some other issues like the possibility of using self-repair strategies, self-reconfiguration for increased fault-tolerance or system adaptation, inclusion of intrinsic evolvable systems, etc., which could be interesting features for WSN technology.

Among SRAM based dynamically and partially reconfigurable FPGAs, the Spartan 6 family offers the lowest, yet not small at all, technology commercially available today, with sufficient number of resources for the targeted application fields. Spartan-6 partial reconfiguration was originally addressed by [23] and [24], so they match with the imposed requirements.

Partial reconfiguration for enhanced boot-up sequences in Spartan-3 devices was addressed by Hubner and Becker in [25], where they proposed a multi-phase technique for fast boot-up. In our approach we also address this issue, but in order to diminish energy consumption in the reconfiguration phase.

III. Proposed Node Architecture

For the correct understanding of the power management techniques used in our proposed HiReCookie platform (High-Performance Reconfigurable Cookie), is necessary to know some information about the node architecture and features.

WSN nodes include at least four functional blocks: processing, communication, sensing and/or acting, and power supply. In order to have a flexible and modular design, the *Cookie* platform is divided into four different PCBs, each of them covering one of these previous roles. Every layer is connected to their neighbors through vertical connectors. See Figure 3. It is possible to exchange every layer separately if

different sensors, communication modules, power supply sources, etc. are needed. This modularity is very useful when adapting the node to different requirements and scenarios. The four layers mentioned are listed below.

Figure 3: Cookie Layer architecture.

- *Sensor layer*: it includes conditioning circuits for both digital and analog sensors and/or actuators.

- *Power supply layer*: The node can be powered from a USB cable, lithium or AA batteries or directly from the mains (using the USB connection). It includes a DC to DC converter (TPS650243) to provide the needed current (up to 1,6A) and voltage level. This power management IC provides three highly efficient step-down converters (up to 97% efficiency) that enter in low power mode at light load for maximum efficiency across the widest possible range of load currents, and two LDOs for lower currents. It can also recharge a 500mAh battery in only one hour from the USB connection.

- *Communication layer*: it includes a radio module to communicate data between nodes. There are both ZigBee and Bluetooth versions. In the case of the ZigBee module, different frequencies are available (2.4 GHZ and 868 MHz). The module used along this work is the Telegesis ETRX2 ZigBee module.

- *Processing layer (HiReCookie)*: it is the brain of the platform. It is the layer in charge of processing all the information given by the sensors and the radio module. It includes a Spartan 6 Xilinx FPGA (XC6SLX150-2) and a tiny microcontroller (ATtiny 2313V) in charge of the execution of the power management tasks together with the necessary tools to implement the proposed power management strategies.

Even though there is a microprocessor in the board, this architecture is not considered as a mixed uP+FPGA system, since the controller is just for power management, and it does not handle nor manipulates any other data. Moreover, the SoPC approach is followed in order to allocate both SW and HW resources inside the FPGA.

The block diagram of the HiReCookie architecture is shown in Figure 4 is divided into two different areas separated by the vertical bus. The components included on the right side belong to the processing layer while the blocks placed on the left side correspond to other layers. The architecture of the processing layer is divided into seven different power islands that can be powered on and off separately. Every one of these islands, together with the power management policies will be explained in the next section. In Figure 4, power islands are represented by colors.

The FPGA is the brain of the platform. It will be in charge of all the processing tasks and management decisions. However, the external microcontroller remains always powered. It works as a sentry to wake up the rest of the system according to the commands given by the FPGA.. In order to achieve autonomous boot-up, an initial bitstream is stored in the Flash memory so that it is automatically loaded into the FPGA once both devices are powered. The Flash memory also works as a storage device for programs, other bitstreams or application data.

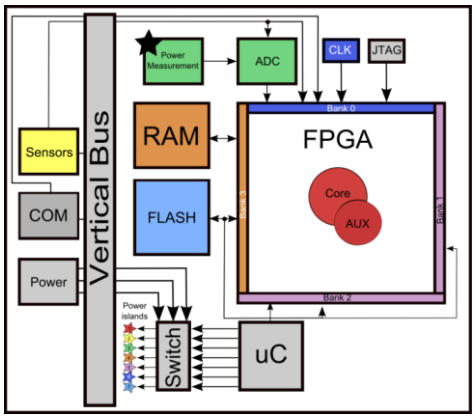

Figure 4: Processing Layer Architecture

The RAM memory is mainly used as an extension program memory to provide fast access to the program data. Since configuration energy is related to reconfiguration speed, this memory may be very convenient when dynamic and partial reconfiguration (DPR) is used.

An ADC converter is also used for the system to be able to process data from analog sensors as well as for measuring the instant current consumption in every island. In this way, the node is power-aware and can take dynamic power management decisions.

IV. LOW POWER ORIENTED ARCHITECTURE

The inclusion of high performance components in the platform leads to high energy consumption. Even though some of the components have their own power save modes, the static consumption is still too high according to the WSN standard levels (in the case of the FPGA is around 60-80 mA). In order to solve this problem, the architecture of the platform has been divided into seven different power islands that can be switched on and off independently in case they are not required. All the different power islands and the reasons why they were selected are detailed below:

- **Island 1**: FPGA core@1.2 V, and auxiliary logic of the FPGA@ 2.5 V
- **Island 2:** Sensor board @ 3.3 V
- **Island 3:** ADC, power consumption circuitry @ 3.3 V
- **Island 4:** RAM memory and FPGA bank 3 @ 1.8 V
- **Island 5:** FPGA banks 1 and 2 @ 1.8 V
- **Island 6:** External clock and FPGA bank 0 @ 3.3 V
- **Island 7:** Flash memory @ 1.8 V

As it can be seen, four different power supply voltages are required: 1.2 V, 1.8 V, 2.5 V and 3.3 V. The FPGA core is the only one powered at 1.2 V and, since it needs to be powered

together with the auxiliary logic, both rails are considered as the same island but not at the same voltage. The auxiliary logic must be powered at 2.5 V. The 1.8 V supply rail is used to power banks 1, 2 and 3, the external memories and the external microcontroller. The external microcontroller is not included in any island since it needs to be powered at all times. The pins used for the communication between the ATtiny and the FPGA are located in bank 2. Apart from that, all the dedicated pins for configuration are shared between banks 1 and 2, so these two input output banks belong to the same island. The RAM memory and bank 3 are placed together in a different island, because the RAM memory controller, which is a hard IP of the FPGA, is placed on the left side where this bank is located. Regarding the 3.3 V rail, independent islands have been included to allow managing sensors separately from the power measurement circuitry. The management of these power islands defines different power down modes that will be discussed in the next section.

In order for the system to be able to wake up from these sleep modes, there must be a component acting as a sentry to manage the wake up signals. This component must be smart enough to be able to handle the power management execution but yet simple enough to have very low power consumption since it is powered at all times. As it was mentioned before, the component selected to carry out this duty is the ATtiny 2313V AVR microcontroller. This controller includes three different power modes: Idle, Power Down and Standby. The Standby mode will not be used since it requires an external oscillator that is not included in this platform. The microcontroller can wake the system up using different sources of interruption. The selection of these sources depends on the power mode that is being used within the ATtiny so that it defines how deeply the system is sleeping.

The microcontroller can work in any of the following low power modes:

- *Idle mode* (10 µA at 0.1 MHz): In this mode, an interrupt coming from its UART, which is connected to the radio device,, any internal timer, a threshold value in an analog sensor or an interruption caused by the FPGA (if it is awake), can wake the microcontroller up.

- *Power down mode* (< 0.1 µA at 32 kHz): In this mode, only an interruption caused by the FPGA and the watchdog timer can wake up the microcontroller. When the node is working inside the Sleep region, the FPGA is not powered so the only way to wake up the microcontroller is the watchdog timer periodically.

Therefore the sources of interruption to wake the system up are: the radio module, analog sensors or internal timers. The power supply layer includes a DC to DC converter that could be enabled or disabled by external signals, but this feature is not being used in this design, since the support provided by the decisions taken by the FPGA and the control on the power islands is more efficient. The IC converter also includes a power save mode that can be either selected externally or automatically, depending on the load. In this way, if most of the islands are switched off, the load is reduced and then the converter enters into power down mode. Apart from the ATtiny and the power supply module, the radio module should be also powered at all times. The radio module includes four different power modes as shown in Table 1. Every one of these modes has different power consumption that depends on whether the node is working either as a router or as a coordinator, or if the node is working as an end-device.

Power Mode	Router or Coordinator	End Device
Awake	36 mA	9 mA
Idle	32 mA	4.5 mA
Asleep 1	0.7 mA	0.7 mA
Asleep 2	0.7 µA	0.7 µA

Table 1: Power modes in the ETRX2 module

Depending on which power down mode is being used in the radio device, there are different ways to wake the module up. When the module is in Asleep 1, it can be woken up through AT commands sent by the ATtiny. However, the power consumption during this mode is not affordable in all cases. When the module is working in Asleep 2, which is the deepest power mode, it can be only woken up using an external interrupt which is also given by the ATtiny controller.

It is important to highlight that even though the ATtiny is the element in charge of executing the power management tasks, the decision of which technique should be used is a competency of the FPGA. In this way, every time the FPGA enters into a sleep mode, it sends a command via SPI to the ATtiny microcontroller with the information of which methodology is going to be applied. Therefore, since the FPGA may be aware of the power consumption of the platform, the way these methodologies are handled can be determined by the application and improved.

V. POWER MANAGEMENT TECHNIQUES

Three different phases can be identified in the lifecycle on an SRAM FPGA based node, as shown in Figure 5. The methodologies to be implemented in every one of these periods are different, addressing different aspects in each phase.

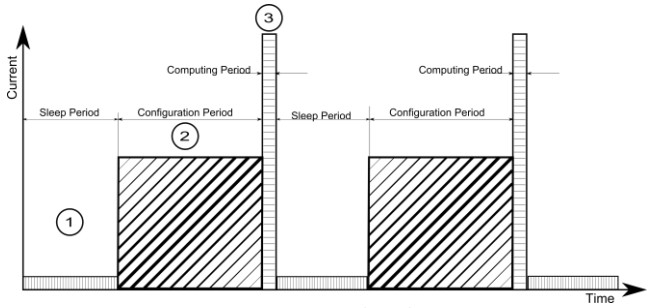

Figure 5: Power consumption profile of a typical application

During sleep period (1), current consumption must be reduced as much as possible and, in this case, the management of power islands and wake up possibilities is crucial. During the FPGA configuration period (2), current consumption is approximately constant, with small changes depending on the frequency of the configuration clock. Therefore, the main goal within this period will be reducing time which, as it will be shown, is directly related to the reduction in size of the bitstream file. Finally, the main constraint during the execution period (3) is the high current consumption during execution.

The main idea in this stage is to compensate the high current with very fast calculations, so that the energy is finally reduced. It is in this period where taking advantage of HW acceleration and parallel HW can make a huge difference compared to SW based solutions.

A. Sleep Period.

During this period, some sleep modes and wake up policies have been implemented to reduce power consumption during inactive periods of time. Even though there are a many different combinations, those ones listed here are the most representative. A summary is provided in Table 2: HiReCookie Sleep Mode.

Sleep Mode 1: This is the deepest sleep mode. The ATtiny can be woken up using only its watchdog timer. Then, for example, it could power the communication module and the sensor layer together in case a message from the radio or any sensor threshold is reached. In case none of these events occur, the ATtiny can go back to power down mode.

Sleep Mode 2: In this mode, the ATtiny is in power down mode, so only the watchdog timer can wake it up. Nevertheless, it is possible to be working in a very deep sleep mode while the sensor layer is not powered. Once the ATtiny is woken up by its timer, it can send an interruption to the communication module to wake it up or it can power the sensors island to check if a threshold value has been crossed. In this case, the communication module can be set also to be always powered.

Sleep Mode 3: In this mode, the ATtiny is also working in power down mode. When the timer wakes it up, it can change to idle mode in order to check if any sensor has crossed the threshold value. This mode only makes sense if the sensor response is critical due to a dangerous parameter where it is not possible to wake the sensor island up and wait until the sensor measurement is stable. A variation of this method is the same configuration but the ATtiny working in idle state in order to have instant response in case a measurement problem occurs.

Sleep Mode 4: This case is a combination of the previous two modes. So, it is possible to wake the system up using the sensors response or a possible message coming from the radio.

Sleep Mode 5: This sleep mode can be useful if a faster response is required since it is not necessary to wake up the microcontroller.

Once the system leaves the sleep mode there are many possibilities that depend on the application. At some point during the execution phase, the FPGA may send new sleep and wake up policies to the AtTiny. As it may be seen in the results table, a careful design and component selection, together with this variety of sleep modes may yield results regarding current consumption during sleep mode below 2 µA in most cases. For a 500 mAh thin flat battery, it would allow over 25 years operation in this sleep mode (with an ideal battery).

B. Configuration time

As it was mentioned before, there are two different methodologies to decrease power consumption during this period: increasing the configuration clock frequency and reducing the size of the bitstream file. It is crucial for the system to include a fast configuration method to automatically load the bitstream into the FPGA every time the system is powered. The configuration method selected is the Master BPI configuration mode. It consists on a parallel connection between the FPGA and a Nor-Flash memory, which may be driven by the FPGA natively at boot-up or by a device installed later, for DPR.. In this way, every time the system is powered, the FPGA starts to generate addresses to read the bitstream file from the external memory. At the beginning the configuration frequency is set to 1 MHz by default. Then, when the FPGA starts to read the bitstream file, this frequency can be changed by editing the header of the configuration file. This frequency can be selected among different values. Even though, the maximum frequency provided by the Xilinx tools is 26 MHz, the maximum frequency achieved for the moment by the HiReCookie platform for configuration is 6 MHz.

The second challenge consists of reducing the size of the configuration file. In order to optimize this reduction, the next three steps were followed: a) relocation of HW modules using the PlanAhead Tool in order to maximize the empty areas, b) compress the bitstream file using the commands given by Xilinx, C) reduce the bitstream file by erasing the empty areas, this is, extracting a partial-initial bitstream rather than using a complete one. Next paragraphs show these steps in more detail.

	Islands							ATtiny	ZigBee	Power Supply Layer	Average Current Consumption	Wake up Possibilities
	1	2	3	4	5	6	7					
Mode 1	-	-	-	-	-	-	-	Power down	OFF	Power down	0.1 µA ATtiny + 1 µA Power	Watchdog
Mode 2	-	-	-	-	-	-	-	Power down	Asleep 2	Power down	0.7 µA Radio + 0.1 µA ATtiny + 1 µA Power	Watchdog Radio
Mode 3	-	●	-	-	-	-	-	Power down/ Idle	OFF	Power down	Sensor layer consumption + 0.1/10 µA (ATtiny) +1 µA Power	Watchdog Analog comparator
Mode 4	-	●	-	-	-	-	-	Power down	Asleep 2	Power down	Sensor layer consumption + 0.7 µA Radio + 0.1 µA ATtiny + 1 µA Power	Watchdog Analog comparator Radio
Mode 5	-	-	-	-	-	-	-	Idle	Asleep 2	Power down	0.7 µA Radio + 10 µA ATtiny + 1 µA Power	Watchdog Analog comparator

Table 2: HiReCookie Sleep Mode

a) The more compact the desing is, the bigger the empty areas of the FPGA internal architecture. The default configuration of the FPGA is composed by zeros. Thus, since the extraction of the partial-initial bitstream consists of erasing those areas filled with zeros, the bigger these areas, the bigger the information to erase.

b) Xilinx gives the possibility of reducing the bitstream size through the use of the –g compress command. This method consists of using the Multiframe Write register (MFW) to indicate that the same frame must be written along subsequent positions of the bitstream.

c) In order to achieve an optimal reduction of the bitstream file, it is crucial to have a deep knowledge about its format and the internal architecture of the FPGA. The internal architecture of the Spartan 6 LX150 is divided in 12 clock regions, 64 CLB columns, 6 BRAM columns, several I/O blocks in each bank, 4 DSP columns, 1 DCM column, the Bus SCAN module, the ICAP module, MDM module, 2 MCBs, etc. In order to edit the bitstream file to erase the empty areas, a SW tool has been developed. The interface is shown in Figure 6.

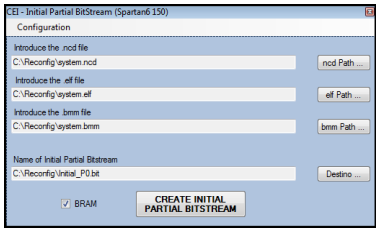

Figure 6: SW Tool to extract the Partial-Initial Bitstream

The structure of the bitstream is shown in Figure 7. Regarding the header section, the only edited fields are the configuration frequency and CRC errors. The configuration frequency will be set to 2 MHz or 6 MHz within the different tests, while the CRC errors are disabled in all of them. Regarding the CLBs and BRAM sections, this is where the maximum reduction is achieved. As it can be seen in Figure 7, the idea is erasing the empty frames keeping the correct address values for the rest of non-empty parts.

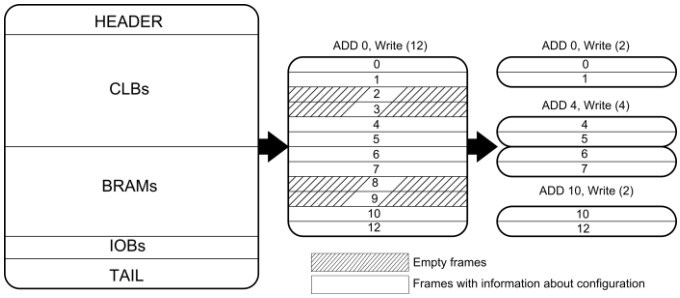

Figure 7: Extraction methodology

C. *Computing time*

In [1], the authors studied the improvement of HW acceleration compared to SW based solutions in the HiReCookie platform, where it was shown that, for high performance applications, HW acceleration may compensate the energy required for configuration or reconfiguration. Results were shown for the case of encryption algorithms, and it was also shown that the tendency to more complex applications and higher performance are in favor of FPGA based architectures much better than more complex processors. So, for this paper, the contributions are then focused in the reduction of configuration time since it was demonstrated in the previous work that it is critical effect to the total power consumption, as it is shown in Figure 8, taken from [1]. Figure 8 shows the working profile of the FPGA core. While the only way to reduce consumption for the rest of the elements is power turn off, in the case of the FPGA the way to reduce consumption is reducing configuration time since it is the bigger variable contribution.

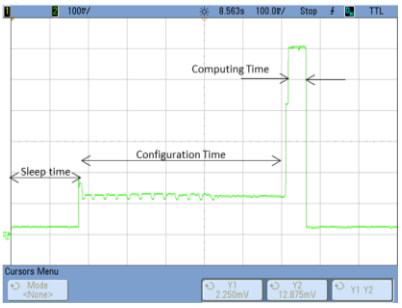

Figure 8: Typical application energy profile

VI. TESTS AND RESULTS

Intensive tests have been carried out in order to evaluate the effect of the reduction of the bitstream files and the frequency of configuration. In this work, the result of four of these tests are shown and compared to see the reduction in terms of time, which is translated in a reduction in the charge consumed by the FPGA core at the beginning of every cycle. In this case, the HW block contained in the bitstream is a Microblaze controller executing a simple task, since it is not the aim of this test to show the advantages of using HW calculations. The charge consumed by the node is also shown in each test. In order to obtain measurements of a real application during operation, the node is working with a ZigBee module in active mode. To obtain the charge consumption for the worst case, all the power islands are switched on. Therefore, all the components are powered including the ZigBee layer and the power supply layer.

A. *Case 1: Total bitstream with no optimization in area and no compression. Configuration clock running at 2 MHz.*

As it can be seen in Figure 9, if no changes are done to reduce the size of the bitstream file, configuration time, 1.04 s, is much bigger than the average computing time of a normal application that can be approximately running for 50 ms. In this way, taking into account that the normal current consumption of the FPGA core during configuration is around 53 mA, this time is the biggest contribution to the total consumption. Due to this reason, reducing this time is a must to achieve an efficient solution. The total node consumption is shown in Figure 10.

B. *Case 2: Partial-Initial bitstream included the optimization in area. Configuration clock running at 2 MHz.*

In this second case, even though the configuration frequency is still 2 MHz by creating the partial- initial

bitstream which is 8.2 times smaller than the total one, a reduction of 7.9 times of charge consumption is achieved.

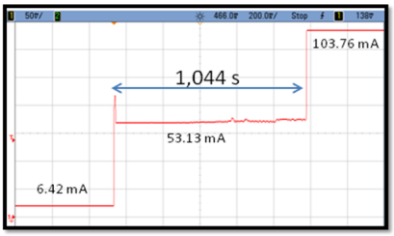

Figure 9: FPGA Core consumption in case 1

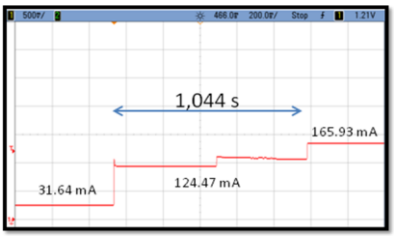

Figure 10: Total node consumption in case 1

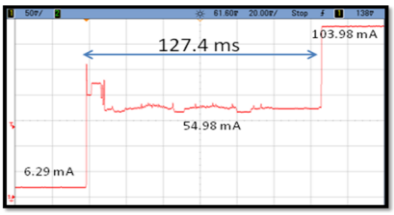

Figure 11: FPGA Core consumption in case 2

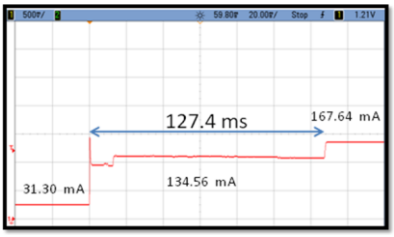

Figure 12: Total node consumption in case 2

C. Case 3: Compressed bitstream but not partial, including optimization in area. Configuration clock running at 6 MHz.

This case analysis is the best that could be achieved using the tools provided by Xilinx. The result is shown in Figure 13. In this case the bitstream is compressed using the -g compress command and configuration is done at 6 MHz.

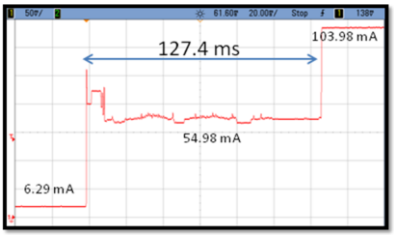

Figure 13: FPGA Core consumption in case 3

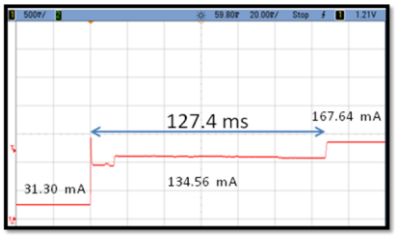

Figure 14: Total node consumption in case 3

D. Case 4: Partial-Initial bitstream with the optimization in area. Configuration clock running at 6 MHz.

The last result shown in Figure 15 represents the best case combining both the increment of configuration frequency and the creation of the Partial-Initial bitstream. In this case, keeping the same frequency value as in case 3, a reduction of 2.4 times of charge consumption is achieved. So, it can be summarized that a 20.22 times reduction can be achieved compared to a non-carefully analyzed reconfiguration. However, for the sake of correctness, it is fair to say that the contribution of the tool presented in this paper reduces 2.4 times compared with what just can be achieved using only the Xilinx Tools.

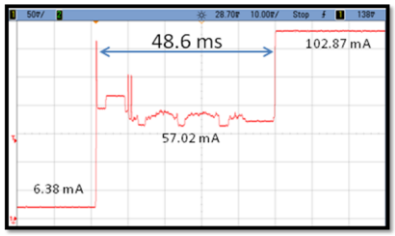

Figure 15: Core consumption in case 4

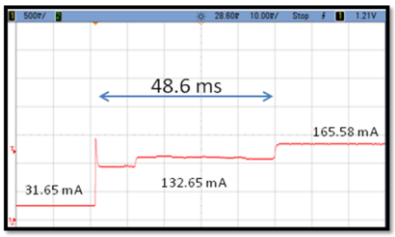

Figure 16: Total node consumption in case 4

Table 3 shows the results of the previous tests, and a calculation of the charge required for every case. It is important to notice that with a flat 500 mAh battery, there would be sufficient energy for almost 279,000 reconfiguration cycles in the best case (not accounting FPGA degradation).

Tests	Bitstream size (kB)	Freq. (MHz)	Time (ms)	Config. Current (mA)	Computing Current (mA)	Config. Charge (µAh)
1	4122	2	1044	53	103	15.37
2	469	2	127	55	104	1.94
3	1352	6	122	54	102	1.83
4	469	6	48	57	103	0.76

Table 3: Final Results

VII. CONCLUSIONS AND FUTURE WORK

We have presented an architecture and a set of power management policies that let a wireless sensor node overcome

the consumption problems posed by the use of an SRAM based FPGA as the main processing element of the node. This is achieved by a carefully designed set of sleep modes which let the node to consume negligible energy during sleep modes, and by minimizing reconfiguration energy, mostly by making it as fast as possible, and reducing bitstream size. DPR has been used to minimize the bitstream size using a custom tool that produces smaller partial bitstreams. Also, it opens the possibility of achieving small energy full boot-up sequences by using multi-stage boot sequences. DPR is also an opportunity to improve HW execution of modules, with the same flexibility of SW applications, but with better energy utilization, as it has been shown in previous works. It also opens the gate for further research like the incorporation of self-repairing capabilities, self-adaptation, evolvable HW, as well as a very high flexibility for deployment and commissioning of WSN systems with nodes that may change both their HW and SW. We claim this architecture to be considered as a single-device FPGA-based board because, even though there is an external ultra-low energy microcontroller, the ATtiny, it does not participate in any processing. Actually, we consider that the System on Programmable Chip approach (SoPC), combined with an internal architecture that lets DPR be used to exchange HW blocks at run-time is a promising technology for High Performance Applications.

ACKNOWLEDGMENTS

This work has been partially funded by the ARTEMIS JTI Project SMART (100032) and the EuroStars Project RUNNER (e!5527)

REFERENCES

[1] Valverde, J.; Otero, A.; Lopez, M.; Portilla, J.; de la Torre, E.; Riesgo, T. Using SRAM Based FPGAs for Power-Aware High Performance Wireless Sensor Networks. Sensors 2012, vol 12(3), pp. 2667-2692

[2] B. Harjito, S. Han, "Wireless Multimedia Sensor Networks Applications and Security Challenges", in International Conference on Broadband, Wireless Computing, Communication and Applications, pp. 842-846, Fukuoka, Japan, November. 2010

[3] I.F. Akyildiz, T. Melodia, K.R. Chowdhury, "Wireless Multimedia Sensor Networks: Applications and Testbeds", in Proceedings of the IEEE 2008, vol. 96, nº10, pp. 1588-1605

[4] T. He, S. Krishnamurthy, L. Luo, T. Yan, L. Gu, R. Stoleru, G. Zhou, Q. Cao, A. Vicaire, J. A. Stankovic, T. F. Abdelzaher, J. Hui, B. Krogh, "VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance", in ACM Transactions on Sensor Networks, vol. 2, Nº 1, pp. 1-38, 2006

[5] Y. Tseng, Y. Wang, K. Cheng, Y. Hsieh, "iMouse: An Integrated Mobile Surveillance and Wireless Sensor System", in Computer, vol. 40, Nº 6, pp. 60-66, 2007

[6] D. Wu, S. Ci, H. Luo, Y. Ye, H. Wang, "Video Surveillance Over Wireless Sensor and Actuator Networks Using Active Cameras", in IEEE Transactions on Automatic Control, vol. 56, Nº 10, pp. 2467-2472, 2011

[7] P. Kulkarni, D. Ganesan, P. Shenoy, Q. Lu, "SensEye: a Multi-tier Camera Sensor Network", in Proceedings of the 13th annual ACM International Conference on Multimedia, pp. 229-238, New York, USA, November 2006

[8] M. Avvenuti, C. Baker, J. Light, D. Tulpan, A. Vecchio, "Non-intrusive Patient Monitoring of Alzheimer's Disease Subjects Using Wireless Sensor Networks", in Proceedings of the World Congress on Privacy, Security, Trust and the Management of e-Business, pp. 161-165, Washington DC, USA, 2009

[9] M. Marzencki, P. Lin, T. Cho, J. Guo, B. Ngai, B. Kaminska, "Remote Health, Activity, and Asset Monitoring with Wireless Sensor Networks", in IEEE International Conference on e-Health Networking Applications and Services, pp. 98-101, Columbia, MU, USA, June 2011

[10] L.A. Grieco, G. Boggia, S. Sicari, P. Colombo, "Secure Wireless Multimedia Sensor Networks: A Survey", in International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, pp. 194-201, Sliema, Malta, 2009

[11] W. Yong, G. Attebury, B. A Ramamurthy, "Survey of Security Issues in Wireless Sensor Networks", in IEEE Communications Surveys & Tutorials, vol. 8, pp. 2-23, 2006

[12] B. Stelte, "Toward development of high secure Sensor Network Nodes using an FPGA-based Architecture", in International Wire & Cable Symposium, pp. 539-543, Istanbul, Turkey, 2010

[13] J. Portilla, A. Otero, E. de la Torre, O. Stecklina, S. Peter, P. Langendorfer, "Adaptable Security in Wireless Sensor Networks by Using Reconfigurable ECC Hardware Coprocessors", International Journal of Distributed Sensor Networks (IJDSN), 2010

[14] V. Cantoni, L. Lombardi, P. Lombardi, "Challenges for Data Mining in Distributed Sensor Networks", in International Conference on Pattern Recognition, vol. 1, pp. 1000-1007, 2006

[15] S. Rup, R. Dash, N.K. Ray, B. Majhi, "Recent advances in distributed video coding", in IEEE International Conference on Computer Science and Information Technology, pp. 130-135, Beijing, China, 2009

[16] T. Ji-gang, Z. Zhen-xin, S. Qing-lin, C. Zeng-qiang, "Design of Wireless Sensor Network Node with Hyperchaos Encryption Based on FPGA", in International Workshop on Chaos-Fractals Theories and Applications, pp. 190-194, China, 2009

[17] G. Chalivendra, R. Srinivasan, "Murthy, N.S. FPGA based re-configurable wireless sensor network protocol", in International Conference on Electronic Design, pp. 1-4, Penang, Malaysia, 2008

[18] P. Muralidhar, P. Rao, "Reconfigurable wireless sensor network node based on Nios core", Fourth International Conference on Wireless Communication and Sensor Networks, pp. 67-72, Allahabad, India, 2008

[19] S. Yan, L. Le, L. Hong, "Design of FPGA-Based Multimedia Node for WSN", 7th International Conference on Wireless Communications, Networking and Mobile Computing, pp. 1-5, Wuhan, China, 2011

[20] Z. Chao Hu, P. Liu Yingzi, Z. Zhenxing, M.Q.H. Meng, "A novel FPGA-based wireless vision sensor node", IEEE International Conference on Automation and Logistics, pp. 841-846, Shenyang, China, 2009

[21] S. J. Bellis, K. Delaney, B. O'Flynn, J. Barton, K. M. Razeeb, C. O'Mathuna, "Development of field programmable modular wireless sensor network nodes for ambient systems", in Computer Communication, vol. 28, pp. 1531-1544, 2005

[22] F. Philipp, M. Glesner, "Mechanisms and Architecture for the Dynamic Reconfiguration of an Advanced Wireless Sensor Node", in International Conference on Field Programmable Logic and Applications, pp. 396-398, Chania, Crete, Greece, 2011

[23] Dirk Koch, Christian Beckhoff and Jim Torresen, Demo Paper: Advanced Partial Run-time Reconfiguration on Spartan-6 FPGAs, Proceedings of the IEEE International Conference on Field-Programmable Technology (ICFPT'10), IEEE, Beijing, China, 2010

[24] Otero, A.; LLinás, M.; Lombardo, M. L.; Portilla, J.; de la Torre, E.; Riesgo, T.; "Cost and energy efficient reconfigurable embedded platform using Spartan-6" Proceedings of the SPIE 2011, vol. 8067, April 2011

[25] M. Hübner, J. Meyer, O. Sander, "Fast Sequential FPGA Startup based on Partial and Dynamic Reconfiguration", in IEEE Computer Society Annual Symposium on VLSI, Lixouri, Kefalonia, Greece, pp. 190-194, July 2010